PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: WO 98/38585

(43) International Publication Date: 3 September 1998 (03.09.98)

(21) International Application Number: PCT/AU98/00116

(22) International Filing Date: 25 February 1998 (25.02.98)

(30) Priority Data:
PO 5274 25 February 1997 (25.02.97) AU

(71) Applicant *(for all designated States except US)*: McLAREN SOFTWARE TECHNOLOGY PTY. LTD. [AU/AU]; 36 Beachway Avenue, Maslin Beach, S.A. 5170 (AU).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: LINGSTADT, Axel [AU/AU]; 36 Beachway Avenue, Maslin Beach, S.A. 5170 (AU).

(74) Agent: MADDERNS; 1st floor, 64 Hindmarsh Square, Adelaide, S.A. 5000 (AU).

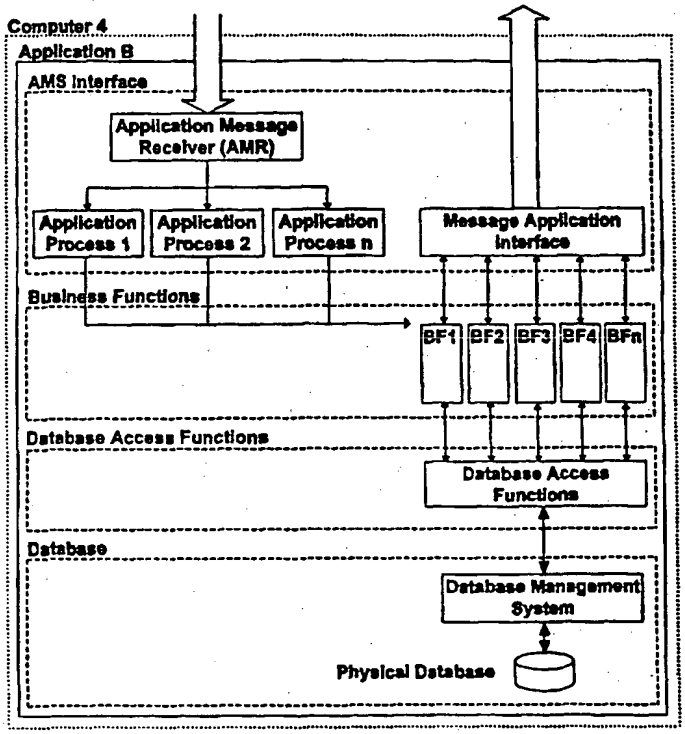
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: APPLICATION MESSAGING SYSTEM

(57) Abstract

An application messaging sytem (AMS) (Fig.3) is provided as a means to simplify access to and modification of a server application located on a server by a client application in a client/server computer environment. The AMS comprises a client end application interface (Fig.2) which operates between the client application and the client/server environment to generate predetermined requests. The AMS also comprises a server end application interface to operate between the client/server environment and the server application to receive the requests. A further element of the AMS is a Business Function module in the server end application interface which translates the received request into generic data identifiers. Those data identifiers operate the server application and also receive the results of the request and pass them in a predetermined format to the client end application interface for presentation to the user as result data. The AMS can be arranged as a first Client/Server database access mechanism which may itself access (Fig.2), via a further AMS, a second Client/Server database access mechanism, and so on as required. The AMS may also be used to control and access a Programmable Logic Controller in a process control environment. The AMS is modular and easily developed, modified and added to without affecting its functionality or ability to work with older versions of requests and replies.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

APPLICATION MESSAGING SYSTEM

This invention relates to Client/Server computer systems and in particular to a method and means for simplifying access to and modification of information contained in an application located on a Server by a Client application or user.

BACKGROUND

In a Client/Server environment most database applications have a resident program which translates the generally natural language query of the user at the Client terminal into a query language statement such as for example a SQL statement. The SQL statement generated by the Client end of the Client/Server program is sent via the network to the Server for processing by the database. The Client program must have detailed knowledge of the database and in particular its tables and attributes. Problems arise if the database administrator changes the database. For example, the administrator may change the table structure of the database which then necessitates that every Client program be modified to accommodate the change. Typically also the SQL statement generated at the Client is changed so that it remains appropriate to the changed tables at the Server.

Thus it can be seen that Client applications have a dependency on not only their particular environment but also on the remote database at the Server. This built-in SQL support function and the Graphical User Interface (GUI) through which the user interfaces with the SQL code at the Client end, are preferably provided by the respective database manufacturer because of this interdependency. However, this increases support costs by requiring program maintenance of the Client program by database programmers who are skilled both in the Server database as well as the Client GUI environment. Costs are also affected by the manner in which these applications are developed since it is typically very difficult to program completely independently blocks of operative code hence testing is on a large scale and very dependent on all the code performing at the one time.

It is also a characteristic of Client/Server systems that they are developed for only one type of Client Application which communicates with only one type of Server Application. Any other Client Application which is required to communicate with an existing Server Application must typically be extensively revised in order to effectively interact with the existing Server Application.

It is further a characteristic of Client/Server systems that any maintenance work on either the Client or Server Application requires the Application to be re-compiled and re-linked.

Furthermore users are only allowed to run one version of the Client Application, which means that every time a modification is made to the Client software or the Server Application every Client must simultaneously receive the updated version of the software.

Yet further, because of the very great interdependency of the code written for the application, even simple requests for information by the user can take a long time to be processed because of the intensive use of administrative code.

These preceding problems contribute to the cost of purchasing and maintaining modern Client/Server Applications.

BRIEF DESCRIPTION OF THE INVENTION

The following brief description of the invention identifies its major elements which are depicted in Figures 2 and 3. A more detailed description of an embodiment of each of the elements follows.

In its broadest form the invention comprises a collection of software modules some resident on a user's PC and some resident on the Server arranged so that a well defined interrelationship between the various modules simplifies access by the user to the Server Application by providing a flexible and efficient messaging system.

For convenience the invention will be referred to as an Application Messaging System (AMS) and for the purposes of this description will use a database as an example of the Server Application.

The AMS model has two major software modules, each referred to as an Application Interface (AI), one located in the Server and one at the Client.

In an embodiment of an AMS Client / Server system an AMS Client Application Interface sits between the network and the Client Application, where the Client Application is most likely a PC based program which is developed using PC based development tools. The AMS Server Application Interface sits between the network and the Server Application which receives one or more messages from any of the Clients.

A message will typically consist of a request to retrieve or modify data on the typically very large and complex database resident on the Server. The Application Messaging System removes complexity from the Client software by forcing the application designer to create a variety of predefined unique requests. This approach is the first of a number of design decisions which greatly reduces the complexity of a Client Server System using the AMS. The effect of this arrangement is to mask the complexity of the Server Application from the Client Application and a user.

The request is bundled into a message which has a format suitable for communication across the Network without the need for the Client or the Server Applications to know anything of Network protocols.

Each request is one of a number of predetermined requests designed by the Server Application developer and the code associated with it is loaded at the Server end. During the lifetime of the Application different requests and their associated Application code can be added as well as modified. In particular, any modified

request becomes a further version of the original request and each of the old and new version/s of the request are useable independent of each other.

The Server AMS Application Interface (Fig 3) receives each request and initially performs various administrative and traffic control operations on the message. Firstly the Server AMS Application Interface applies the request to a validity check and then to one of a plurality of Application Process modules. The Application Process module checks the type of message which is contained within the request and then passes the request onto a predetermined function module according to the type of message it is. This is the point at which the Server end of the AMS hands over the relevant data. The module the message is passed on to is referred to as a Business Function.

The chosen and thus appropriate Business Function reads the request and performs a query of the database via the Database Access Function module which is a part of the Server Application. An example of such a request might be to retrieve all accounts statements for a given account number. The Business Function uses generic data identifiers not database specific data identifiers thereby maintaining a programming buffer between the database and the AMS. The use of generic data identifiers isolates the Business function from inevitable changes in the database and vice versa requiring only the Database Access Function to be re-compiled and re-linked when necessary.

The Business Function also collates the requested data, and because of the predetermined nature of the request and the likely form of the reply, the outgoing format of the data is also predetermined, but of course, the various values of the requested data are specific to the fields being queried.

The Business Function uses the AMS Application Interface to translate the data into a message reply to assist the Client Application or user requesting the data. This reply

is then communicated via the Network back to the AMS Client Application and then presented to the Client Application or user.

In a broad aspect of the invention an application messaging system for operating between a client application and a server application in a client/server environment, comprising,

a) a client end application interface which operates between said client application and said client/server environment to generate one of a plurality of predetermined requests having variable data based on a selection of said request and data by the client application, and

b) a server end application interface which operates between said client/server environment and said server application to receive said request and said variable data for translation into generic data identifiers suitable for operating said server application and for receiving from the server application the results of said request as result data and passing said result data as a reply to said client end application interface for presentation to said client application as a reply to said request.

In a further aspect of the invention the server end application interface further comprises a business function module for each of said predetermined requests each said module having a version identifier which only responds to like versions of said predetermined requests having a respective version identifier; wherein said business function module comprises the use of generic data identifiers to control the operation of said server application and provide said variable data to said server application.

A specific embodiment of the invention will now be described in some further detail with reference to and as illustrated in the accompanying figures. This embodiment is illustrative, and not meant to be restrictive of the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig 1 depicts a typical Client/Server environment;

Fig 2 depicts a simplified block diagram of an Application Messaging System;

Fig 3 depicts a detailed message flow diagram of the AMS Server Application interacting with a database management system on a Server;

Fig 4 depicts steps 1 to 3 of the AMS Server Application;

Fig 5 depicts steps 4 to 6 of the AMS Server Application;

Fig 6 depicts steps 7 to 8 of the AMS Server Application;

Fig 7 depicts step 9 of the AMS Server Application;

Fig 7.1 depicts a variation of step 9 of the AMS Server Application dealing with a PLC access function;

Fig 8 depicts steps 10 to 14 of the AMS Server Application; and

Fig 9 depicts step 15 of the AMS Server Application;

DETAILED DESCRIPTION OF AN EMBODIMENT OF THE INVENTION

Fig 1 depicts the major components of a typical Client/Server environment, comprising a communications Network for interconnecting computers, a Client Application A1 and its associated Client End Application Interface (Client API) running on Computer 1, and a Server Application B and its associated Server End Application Interface (Server API) running on Computer 4.

The functional and technical requirements of Client/Server environments are well known and will not be discussed here in any great detail. However, it will be understood that both the terminology and the typical hardware upon which the environment described herein is based on current knowledge and this should not restrict the manner in which the invention may be applied in the future.

Typically Client/Server environments use Personal Computers (PC's) as the Client computer. A Client Application program controls the way in which the user of the PC interacts with the Server and the program resident on it. Most commonly a Graphical User Interface (GUI) is used to present the Application to the user of the PC. The Server Application is often a database which is preferably centrally located, adapted to allow multiple seemingly simultaneous access from many users, access control and provide adequate back up procedures for the protection of the valuable

data stored in the Server associated with the Server Application which for the purposes of this description is a database but which could be any other type of application..

Fig 1 also depicts the simplest configuration of an AMS Client and Server arrangement where the Client API is the AMS Application Interface and the Server API is the AMS Application Interface; while Fig 2 depicts one possible configuration of AMS Client and Server Applications with added complexity. As depicted, the AMS can be used to allow multiple different AMS Clients to interact using multiple different AMS Server Applications distributed over a network with multiple PC's and Servers. Not shown is the configuration wherein multiple AMS Client Applications are installed on a PC working with multiple AMS Server Applications installed on one or more Servers. The topology disclosed in Fig 2 and discussed above allows a request to be passed to more than one Server Application to obtain the necessary ingredients to populate a reply. Thus a retail database may need to access a wholesaler's database which may need to access a manufacturer's database to provide the required data to fulfil a predetermined user request.

AMS REQUEST/REPLY MESSAGE

AMS Client and Server Applications communicate via the Network using specific request and reply messages designed by the Application developer. Each message sent contains attributes only relevant to the developer's needs to satisfy a particular user request. The technical detail of communicating messages over the Network is not important to the Application developer and Application user.

The type and content of a reply message will vary based on the parameters within the request message. For example, a Client Application may receive different reply messages based on the nature of the request. Furthermore, request and reply messages are version controlled. Version control provides a way for messages to be flexible and more importantly allows more sophisticated messages based on the

latest arrangements to be processed as well as older messages. Version control will be described in greater detail later in the specification.

Over time, as business requirements change new requirements generally require the user to acquire greater quantities of detailed information. The request and reply messages of this embodiment of the invention support new requests without losing the ability to use old requests.

To obtain a high throughput of customer information between Client and Server Application, the AMS has been designed to request and reply with arrays of records in one message.

A message will be created using the message name, method, version number and the number of attributes the message record will contain. The Client Application will then populate the message using the attributes row number and rank number as two-dimensional array indicators. The Client Application (also called a source application) will then send the message to the Server Application (also called a destination application). The Server destination application will read the message and is able to extract the number of records contained in the message and subsequently able to read all its records and attributes. If the Client source application has not populated an attribute within a message record then the Server destination application will receive an indicator, for example "Value is Null", when trying to read the attribute from the message.

This behaviour compresses non-related request data records into one request message, which can then be processed in one sequence by the Server application and sent back as one array of result records.

AMS REQUIRED MESSAGE ATTRIBUTES

Every request and reply message is preferably identified by a unique identifier element comprising preferably its name, method and version. The following are by

way of example only, and are merely preferable for the embodiment described. Other like parameters may be required or others may be created for different situations. Preferably a message in this embodiment comprises the following attributes:

- Message Name

  The message name is a character string
- Method (Type)

  The method is an integer number
- Version Number of the message

  The version number is an integer number.
- Number of Attributes in the message record.

  The number of attributes is an number.

AMS DATA TYPES

The following attributes including data types and data sizes are preferably available in the AMS Request/Reply Message:

- Character String (attribute type)

  A message variable of type Character String is limited to 32,000 bytes.
- Integer (attribute type)

  Integers of 8 bit, 16 bit, 32 bit and 64 bit size can be supported (attribute size). The support of certain integer sizes is largely dependant on the capabilities of the Client/Server computers.
- Float (attribute type)

  Floating Point variables can be supported up to 128 bits in size, but is not restricted if larger sizes should become available. Here again, the data type is dependent on the computer (CPU) performance and architecture.
- Binary (attribute type)

  A binary variable is only limited by the attribute "size". Based on the host CPU instruction "size" the binary value could have an integer size of 64 bits, but is not restricted if larger sizes should become available.

AMS CLIENT APPLICATION

In a typical exchange of messages any request message issued by the Client Application is sent to the Server Application after which the Client Application performs other tasks as required. The Client Application does not need to wait until the reply message is received. In fact after a request message has been sent, the Client Application may immediately issue another request message, independent of the first request. Noting that the AMS Client API merely interfaces between the Client Application and the Network.

The Client Application periodically or on a random basis enquires with the AMS Client API as to whether a reply message has been received. These enquiries are conducted independently of the user, the Client Application GUI and the Server.

AMS SERVER APPLICATION

Referring to Fig 3, the various AMS modules within a Server are depicted, namely the AMS Interface module and a Business Function module. The Database Access Function module is more properly associated directly with the Server Application. The Server also contains a specific Server Application but could also contain more than one Server Application, for the purposes of describing this embodiment, a database type Server Application is used which could be Oracle, Ingres etc. It will also be noted that the database used in the embodiment is a relational database but other types of database could be used (eg Object Oriented, Hierarchal etc).

During the initialisation phase of the Server Application the Application Message Receiver module in the AMS interface loads a volatile distribution table of predetermined and allowable message names, methods and the Application Processes allowed to process message name and method combinations.

The AMS Server API receives request messages from AMS Client API's. The request message is first security checked by the Application Message Receiver before any further processing of the request is commenced. After the security checks have been

successfully completed, the AMS Server Application Interface invokes the appropriate one of a plurality of Business Functions related to the request message attributes through an appropriate Application Process which is determined by the type of request as identifiable by the message name of the request. The appropriate Business Function accepts the request parameters and performs a predetermined set of instructions since it is written specifically for the name, method and version of the request message it receives. Each required Business Function has been designed by the Server Application Developer.

In one example the instructions will slavishly map the request message into generic database access commands based on the content of the message and its attributes. In a simple request, the user may want to retrieve all the account statements for a given account number, whereas a more complex request may involve calculating the annual maintenance cost of a production line. Each Business Function only contains a very specific set of instructions thus simplifying the creation of a Business Function, one for each desired request and later versions thereof.

Consequently the Business Function module returns to the user a reply message containing the requested data but preparation and transmission of the reply message is performed by the Application Messaging System located within the AMS Server Interface while access to the database, via Structured Query Language (SQL) statements for example, is performed by a Database Access Function module. There is only one Database Access Function module to service all of the different Business Functions. Since the Business Function contains no network code and no database specific code, Business Function code can advantageously be compiled and linked, independent of all network, database and other code. SQL is but one of many first level query languages which sit above a database and provide a first level of abstraction above the in depth structure of the specific database used as a Server Application.

The AMS interface includes a control module (not shown) which monitors the performance of the AMR and the Application Processes. The control module also replies to any outstanding request message where the standard means of responding to a request has failed. If, for any reason, the AMR or any of the Application Processors should fail, the control module is tasked to recover these processes to their original state. Details of all the other administrative tasks of the control module are not discussed since they will be appreciated by those skilled in the art.

SECURITY FUNCTIONS OF THE SERVER AMS

Referring now to Fig 4, the Application Messaging Receiver provides security based on information provided with the request. This information can be coordinated with the information provided when the user logs on to the Client PC which is communicated to the Server. Thus the Server not only contains the software to perform network related security checks but also the software to perform checks using the resident Application messaging System modules and in particular the Application Messaging Receiver (Step 1). After the "user" has successfully logged on to their PC and the AMS has been updated with information in relation to the User, the Application Messaging System controls by way of its predefined program (Step 2) what type of attributes the request messages has and that the user is permitted to issue request messages (Step 3).

It should be noted that the "user" could also be an autonomous Client Application (for example a PC acting as an agent for a user, it could also be another Server Application acting on the command of a request received by it). The ability of the AMS to be configured in this way provides flexibility and strength to its application in the real world.

Thus, the AMS is arranged to check every request message sent to the Server Application for validity, by checking the user and the type of request they are permitted to send. A user issuing an unauthorised request message will not receive a reply containing relevant information since their request will be unable pass the

required security checks and therefore will not be able to access the Server Application.

The AMS modules in particular the Application Message Receiver verifies the authentication key of the request message against a volatile authentication key table as described briefly in Step 2 of Fig.4, so as to provide security functionality which goes beyond the usual user name/password model. In certain applications digital certificates could be received in lieu of other types of authentication data.

In a preferred arrangement the AMS modules provide the Server Application with security tables which contain

all user names and passwords
all user groups
all request messages and their methods
all user group related request messages and methods
all user related request messages and methods.

At run-time, the Application administrator will be able to insert/modify/delete

the names of request messages and their methods
the user names and passwords
the user groups
the request messages and methods per user group
the request messages and methods per user

Thus the required AMS modules are able to check the veracity of a logon request message against the contents of the security database. If the check is successful then all request messages and methods the user is permitted to use are inserted into a run-time authorisation table. The Server API issues an authentication key for this newly created session and inserts this authentication key into the logon reply message and into the volatile authentication table.

Subsequent request messages are evaluated against the authentication key and against the messages and methods recorded for this user in the volatile authorisation table (Step 3 of Fig 4). If any of these evaluations fail, the request will be rejected and the user may or may not be advised and an audit log may or may not be kept.

These functions are part of the Server API and are completely transparent to the Application developer and user.

LOGON BY THE CLIENT APPLICATION USER

When the user of a GUI based Client Application on a PC, wishes to retrieve particular data from a Server Application, typically the Client Application requiring access to the Server Application will issue a message to the Server Application. The message preferably contains the Client's user name and password. Note that if required the Network may invoke encryption and decryption as required but this is done transparently by the Network and does not require the AMS Client or Server APIS's to know what is being done.

The Server API receives the message, evaluates the user name and password and sends a reply message to the Client API. The reply message may preferably contain an authentication key to be used for subsequent request messages by that user and Client Application. The authentication key is transparent to the Application developer and Client Application/user. Once the Client Application receives the reply it is able to send various request messages to the Server (Steps 1, 2 and 3 of Fig 4).

THE CLIENT REQUESTS INFORMATION

The Client Application issues a request message RQ 1. This request message RQ 1 has been designed by the Application developer. In broad terms the Client Application fills or populates the attributes requested in the message RQ 1 and then sends the message, via the AMS Client API, to the AMS Server API. The information provided may be the account number of a customer or the contact details of a new

customer. The AMS specific attributes of the request message such as Authentication Key and User identity are filled in by the AMS Client API without interaction with the Client Application or the Client Application/user.

CLIENT APPLICATION DEVELOPMENT

The Developer of the Client Application can select the development tool of their choice to present data and request query data collection fields in the most appropriate way to the Client Application or user of the Client Application. During the development of the Client Application, the layout of the request and reply messages will be established . The Client Application is not concerned with how the data is extracted by the Server Application, or, whether the data is extracted from a different kind of resource such as a Web Site, word processing document field, digital record, digital multimedia field, CD Rom, or other Server Application resident locally or remotely etc. The implementation of data exchange between the AMS Client API and Server API modules is completely hidden from the Client Application. The Client Application initiates a request message and expects a reply message containing the requested information.

Any Application development tool can be used to develop a Client Application. The developer of the Client Application adapts to the predetermined request and reply messages of the AMS Server API and its Business Function Module. The developer merely requests information from the Server by issuing a request message to the Server using the appropriate AMS message.

CLIENT APPLICATION MAINTENANCE

Most likely the Client Application will be implemented using a PC specific Application development tool to take advantage of the rich resources of PC Graphical User Interface (GUI) development tools. The AMS Client API will also contain the User designed messages and code to request and receive messages. Changes to the Client Application (and GUI) only affect the PC on which the change is taking place. Changes to the "user" designed message affect only the Client

Applications which require these changes, and since requests and reply messages are version controlled, other Client Applications which are not affected do not have to change since they are still able to operate using messages having older version numbers. However, each new request will require a new Business Function.

SERVER RECEIVES A REQUEST MESSAGE

Referring again to Fig 4, the AMS Server API receives the request message RQ 1 which firstly decodes the message (Step 1) and then checks the authentication key of message RQ 1 (Step 2). If the authentication key is valid the AMS Server API will check whether this particular user is authorised to issue request message RQ 1 (Step 3). Referring now to Fig 5, if the check is successful, the AMS Application Message Receiver allocates the message to an Application Process (Step 4) which in turn choses a particular Business Function RQ 1 (Step 5) by extracting the message name from the request message.

SERVER ACTIVATES A BUSINESS FUNCTION

Referring to Fig 5, the Business Function associated with RQ 1 will service the request made by message RQ 1 (Step 6). The Business Function has been designed, coded and implemented by the Server Application developer. The Business Function is stand alone code, compiled and linked independent of the Server Application, in this embodiment a database, and the AMS software modules. The Business Function (Step 7 of Fig 6) reads the attributes of request message RQ 1 (Step 8) and processes the data contained in the message (Steps 9 to 9.2 of Fig 7). The Business Function communicates to the Server Application database using Database Access Functions. The Business Function processes its own logic reading and modifying data (Step 9) which will invoke the Database Access Functions (Step 9.1) to translate generic request parameters into DBMS specific statements. In one example and typically so, those statements are written in Structured Query Language (SQL) statements which is a specific example of a first level query language (Step 9.2) used by databases to read, write and modify data within the database.

The Business Function also receives the output from the Database Access Function and thus is able to reply to the request message RQ 1 by issuing a reply message, say RP 1 to the Message Application Interface within the Server AMS API.

THE BUSINESS FUNCTION REPLIES TO A REQUEST

The reply message RP 1 has also been designed by the Application developer. The Business Function instructs the Message Application Interface (MAI) a part of the Server AMS API to create (Step 10 of Fig 8) a reply message to the given request message which is created by the Server AMS Interface by inserting all transport related parameters in the reply message (Step 11 of Fig 8). The reply message RP 1 is populated by the Business Function (Step 12). After the reply message has been populated, the Business Function instructs the Server AMS API to send the reply message to the Client (Step 13) and the Business Function will have then completed its intended task (Step 14). The Server AMS API then sends a reply message to the Client (Step 15 of Fig 9) via the Network.

In a preferred arrangement, the functions of each Business Function module is restricted to the particular requirement of a corresponding request message, which includes:

reading the request message,

processing the request by manipulating data (reading and writing) using database access functions;

instructing the Server AMS API to construct a reply message;

populating the reply message; and

instructing the Server AMS API to send the reply message.

Simple requests will result in simple Business Functions. More complex requests will require more functions to be performed by the Business Function.

However, a Business Function will perform what it has been designed to do and advantageously each Business Function is developed and compiled upon demand and independent of other Business Functions.

The Business Function **does not** perform:

security tasks;

SQL statements;

Client evaluation tasks;

or any other task which is not directly related to the request.

AMS CLIENT RECEIVES A REPLY MESSAGE

The AMS Client Application Message Receiver (Step 1 of Fig 4) receives the reply message related to the original request message RQ 1. The AMS Client API reads the attributes from the reply message RP 1 after which the reply message is deleted.

DATABASE ACCESS FUNCTION DEVELOPMENT

Database Access Functions are the interface between the generic Business Function and the database specific system calls: for example the performance of a specific SQL statement to extract specified data from the database. If any Business Function module requests or modifies data on the database it will call the generic function name in the database access function library acting not unlike a shared library to the Server AMS API and the Database Access Functions. The Database Access Function issues the SQL statements and passes back the information, to the Business Function. The Database Access Functions are designed by the Application developer specifically to operate with the chosen database management system.

The Database Access Functions are compiled and linked independent of any other AMS modules.

Any changes to the database table structure will result in code changes in the database access functions only. The AMS Business Functions are not affected by these changes.

PLC ACCESS FUNCTION DEVELOPMENT

Referring to Fig 7.1 the Programmable Logic Controller (PLC) Access Functions Module is the interface between a particular generic Business Function and the highly specific syntax requirements of a given Programmable Logic Controller. The Business Function will call a generic function name within the PLC access function library and will pass the relevant parameters to the PLC access function. The PLC access function will translate the given parameters into syntax understood by a PLC which will perform the requested task suitably controlled by the PLC. PLCs are used extensively in industrial process control. PLCs can also be the source of information, so that with an appropriate request a reply containing the requested information can be provided via the PLC Access Function. For example, PLCs can easily count the number of operations performed thus the number of products handled by various machinery can be kept track of.

Step 9.3 depicted in Fig 7.1 shows the Business Function processes which involve the request and the manipulation of data associated with the request being translated in a manner suitable to supply appropriate data to the PLC Access Function Module. The PLC Access Function Module at step 9.4 translates the generic parameters and data supplied by the Business Function into the specific format required by the specific PLC being used.

At step 9.5 the PLC responds to the specific instructions or provides the requested data for transfer back to the Business Function module. In some instances the data being returned is of the form of a confirmation that a PLC operation has been performed.

AMS CLIENT/SERVER APPLICATION DEVELOPMENT

The Application Messaging System allows Application developers to design and implement Client and Server Applications independent of each other. Thus, Application developers are able to design messages to be sent between Client and Server which suit their own and the user's requirements. The messages designed by the developer are the only connection between the Client and the Server Application thus there exists a flexibility not normally associated with Client Server systems. When there are changes driven by business restructuring, user needs and customer requirements, the software supporting the Application must change accordingly. One way of accomplishing this goal is to break up the functional requirements of the business into individual components such as those described herein and as used by the AMS model.

The Application Messaging System provides a messaging service where software Applications can be developed as individual components in a time frame which will reduce the development, and maintenance cost of Client/Server Applications compared to current costs. Modified and newly created AMS based Applications will maintain the security attributes of an existing AMS arrangement.

AMS MESSAGE MAINTENANCE

The AMS approach requires the Application Developer to use version numbers for each modification to a message. Based on ever changing business requirements, each modified message changes the version number which allows the original "old" message to continue and operate as and when required at both the Client and Server. Thus Applications will still function using the "old" messages, whereas newer Applications will take advantage of enhanced messages.

DATABASE ACCESS FUNCTIONS MAINTENANCE

Modifications to the database do not affect the Business Functions. The database specific SQL commands are embedded in the Database Access Function module

which is specific to the requirements of the database supplier. The chosen databases management systems operate separately from Business Functions because it is the Database Access Function module which acts as an interface between the Business Function and the database specific coding techniques required to manipulate data in the database as required by the various request messages.

PLC ACCESS FUNCTION MAINTENANCE

If, at a later date, a PLC is to be replaced by a newer model, the required changes to the AMS will occur only in the specific PLC access functions for that machine. The Business Function, being generic, is not affected. Although if newer functions are introduced a newer version of the Business Function could be developed.

Thus, the Application Messaging System described has the following features:

- the Client and Server Applications are independent of each other and therefore can be compiled and linked independently.
- Client Applications are able to continue processing after a request message has been sent even though a reply message has not yet been received.
- Client Applications are able to send request messages independent of previously sent request messages.
- each request message has an independent Business Function which services that request.
- request and reply messages only contain attributes which are determined by the Application developer.
- all request messages carry an internal authentication key which is used by the Server Application to recognise valid request messages.
- request messages and the modules they use are version controlled, which means older messages from older Client/Server applications will still function using a message with an older version while newer Applications are able to use newer messages, with the additional attributes of a newer version request.
- development of the system is much shorter in time in comparison to typical Client/Server applications because business functions can be developed and used

independent of others thus a debtor listing function can be created and used while a debtors overdue business function is still being created.

- changes in the types of requests are easily accommodated by creating newer versions of business functions or creating new ones.
- client applications can expect a reply from a simple request very quickly since the dedicated business function has only one task to complete, the majority of the delay may be caused by the physical delay to pass the request and its reply over the Client/Server environment. More complex requests may take more time.
- a client application may be a server application or a part of a server application of a further Application Messaging System.
- Client and Server Applications are not restricted to serving databases, they may be industrial programmable logic controllers (PLC's); counters; turnstiles; automatic teller machines (ATM's); etc.

It will be appreciated by those skilled in the art, that the invention is not restricted in its use to the particular application described, nor is it restricted to the features of the preferred embodiment described herein. It will be appreciated that various modifications can be made without departing from the principles of the invention, therefore, the invention should be understood to include all such modifications within its scope.

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. An application messaging system for operating between a client application and a server application in a client/server environment, comprising,

a) a client end application interface which operates between said client application and said client/server environment to generate one of a plurality of predetermined requests having variable data based on a selection of said request and data by the client application, and

b) a server end application interface which operates between said client/server environment and said server application to receive said request and said variable data for translation into generic data identifiers suitable for operating said server application and for receiving from the server application the results of said request as result data and passing said result data as a reply to said client end application interface for presentation to said client application as a reply to said request.

2. An application messaging system according to claim 1 wherein said server end application interface further comprises a business function module for each of said predetermined requests each said module having a version identifier which only responds to like versions of said predetermined requests having a respective version identifier; wherein said business function module comprises the use of generic data identifiers to control the operation of said server application and provide said variable data to said server application.

3. An application messaging system according to claim 2 wherein each request and reply further comprises a unique identifier element indicative of the message name and method.

4. An application messaging system according to claim 3 wherein said server end application interface further comprises an application message receiver for checking the validity of each received request by checking that said message name and method part of said unique identifier element is a predetermined message name

and message method part; and passes valid requests to an application process module which determines the name of each request by checking said name part and passing data associated with said request to a respective said business function for processing requests of that name and the data associated therewith.

5. An application messaging system according to claim 4 wherein said server end application interface further comprises a volatile authorisation table containing predetermined message name and message method identifiers against which received request message name and message methods are checked.

6. An application messaging system according to claim 2 wherein said business function module receives a response from said client application to said query comprising data associated with predetermined unique identifiers and a reply for transmission to the client application using a predetermined format based on said request method and said version identifier.

7. An application messaging system according to claim 3 wherein said request further comprises an attribute identifier representative of the quantity of attributes in said request.

8. An application messaging system according to claim 7 wherein each said attribute is represented by identities comprising the attribute type and attribute size.

9. An application messaging system according to claim 1 wherein said client end application interface need not receive a reply to a prior request before generating a further request.

10. An application messaging system according to claim 1 wherein said server application is a data base comprising data records which said client application at a client end in a client server environment accesses using a first level query language associated with said server application.

11. An application messaging system according to claim 10 wherein said server end application interface further comprises a database access function module with which said first level query language operates.

12. An application messaging system according to any preceding claim wherein said client end application interface further comprises an authentication key associated with each said request and said server end application interface further comprises a key authenticator to check the validity of said authentication key so as to allow only valid requests to be processed by said server application.

13. An application messaging system according to claim 4 wherein said server end application interface further comprises a plurality of application process modules for receiving valid requests from said application message receiver, which application process module receives said request based on a volatile distribution table in said application message receive which contains a message name and message method in accordance with a predetermined application process module.

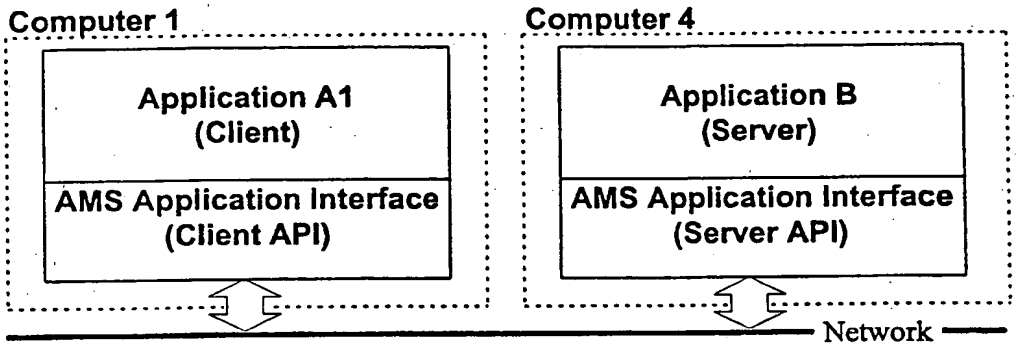

Computer 1
Application A1
(Client)
AMS Application Interface
(Client API)
Computer 4
Application B
(Server)
AMS Application Interface
(Server API)
Network

Figure 1

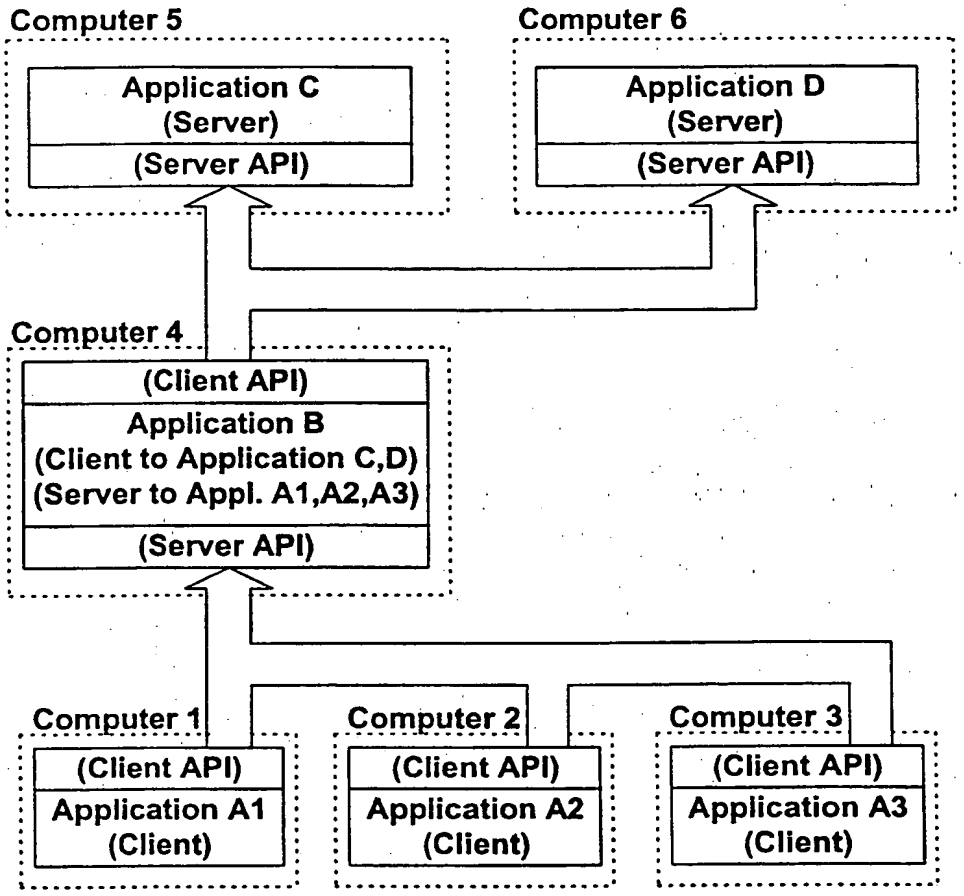

Computer 5
Application C
(Server)
(Server API)
Computer 6
Application D
(Server)
(Server API)
Computer 4
(Client API)
Application B
(Client to Application C,D)
(Server to Appl. A1,A2,A3)
(Server API)
Computer 1
(Client API)
Application A1
(Client)
Computer 2
(Client API)
Application A2
(Client)
Computer 3
(Client API)
Application A3
(Client)

Figure 2

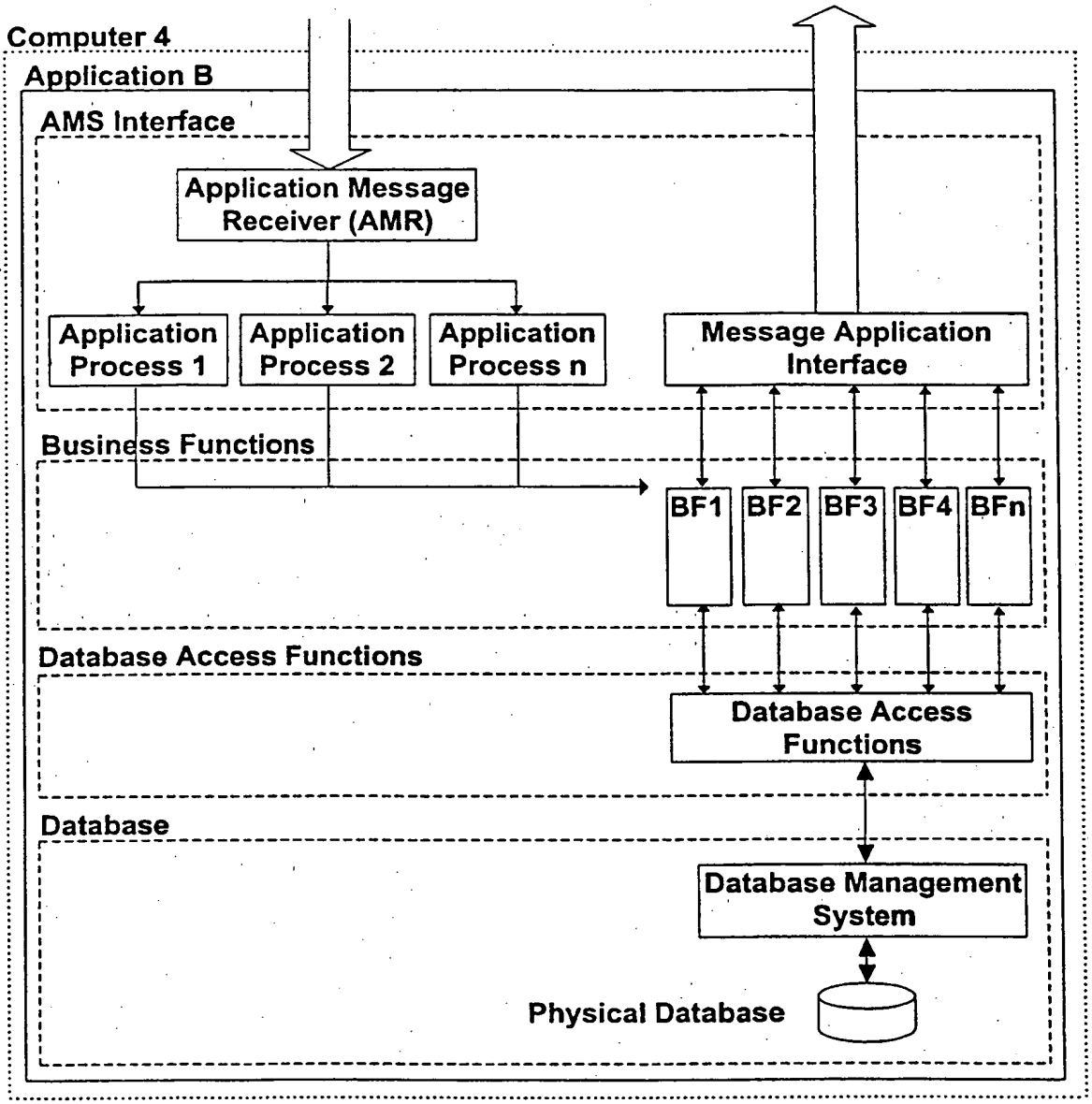

Computer 4
Application B
AMS Interface
Application Message Receiver (AMR)
Application Process 1
Application Process 2
Application Process n
Message Application Interface
Business Functions
BF1
BF2
BF3
BF4
BFn
Database Access Functions
Database Access Functions
Database
Database Management System
Physical Database

Figure 3

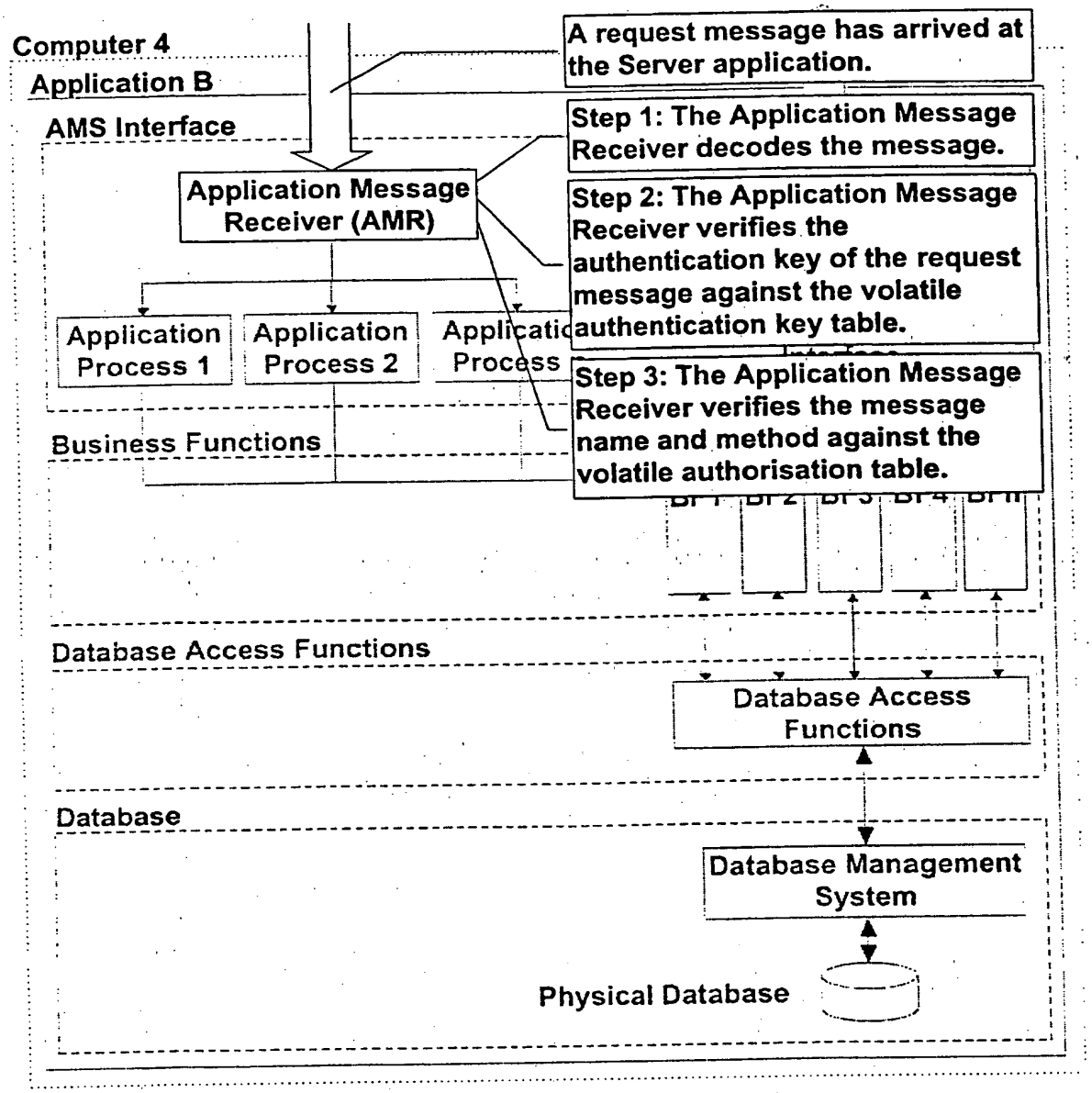

Computer 4
Application B
AMS Interface
A request message has arrived at the Server application.
Step 1: The Application Message Receiver decodes the message.
Application Message Receiver (AMR)
Step 2: The Application Message Receiver verifies the authentication key of the request message against the volatile authentication key table.
Application Process 1
Application Process 2
Application Process
Step 3: The Application Message Receiver verifies the message name and method against the volatile authorisation table.
Business Functions
Database Access Functions
Database Access Functions
Database
Database Management System
Physical Database

Figure 4

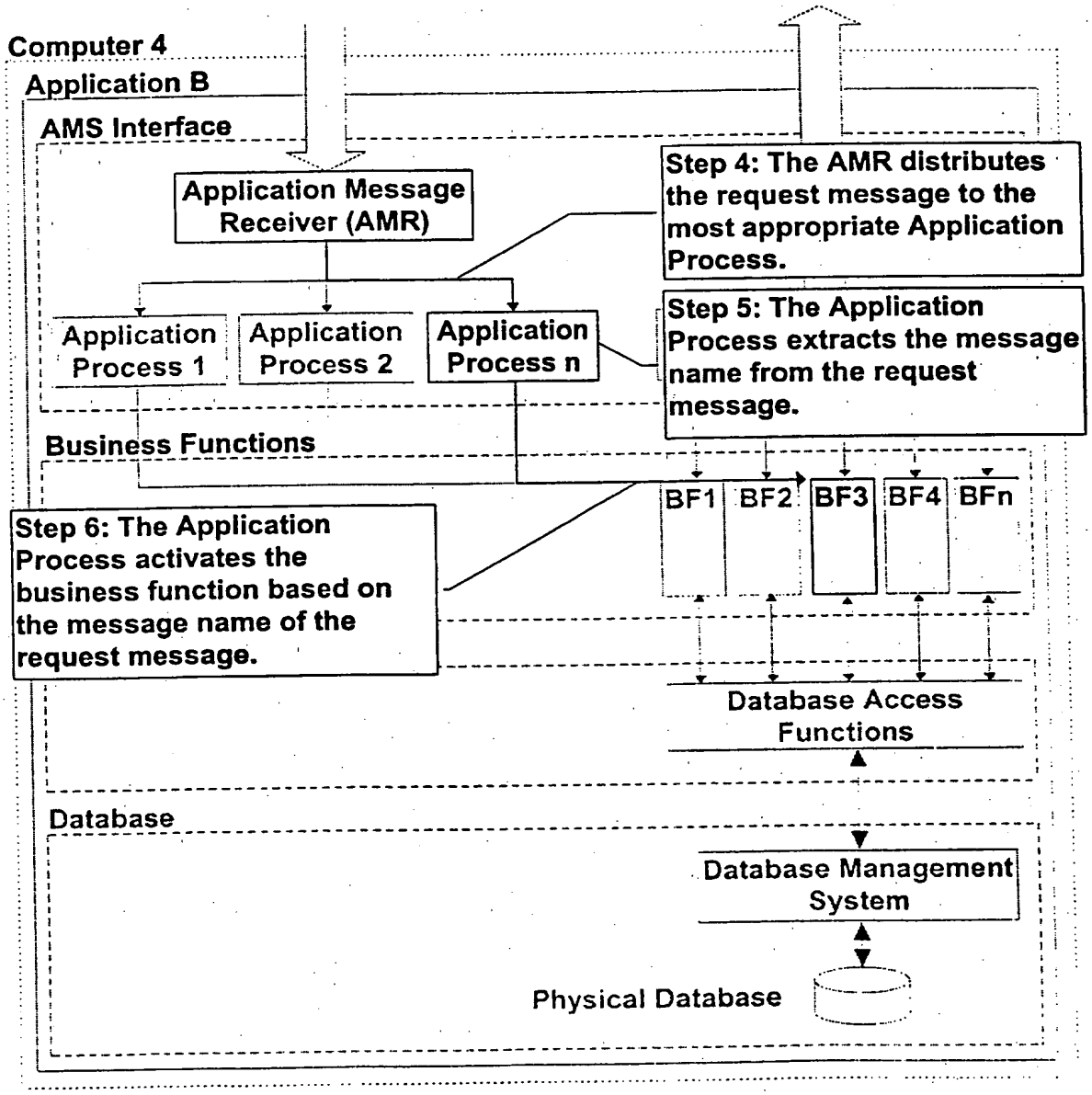

Computer 4
Application B
AMS Interface
Application Message Receiver (AMR)
Step 4: The AMR distributes the request message to the most appropriate Application Process.
Application Process 1
Application Process 2
Application Process n
Step 5: The Application Process extracts the message name from the request message.
Business Functions
BF1
BF2
BF3
BF4
BFn
Step 6: The Application Process activates the business function based on the message name of the request message.
Database Access Functions
Database
Database Management System
Physical Database

Figure 5

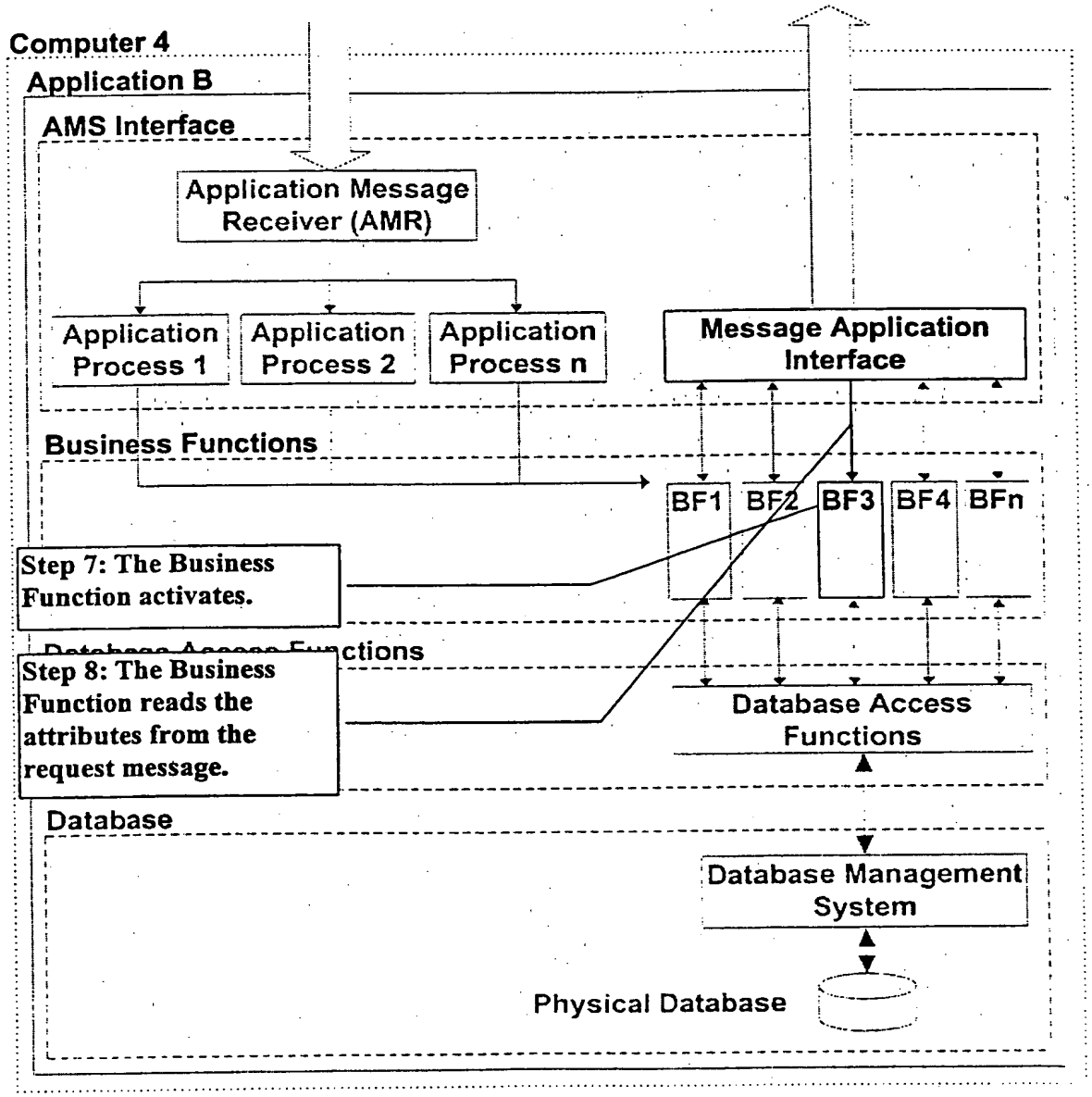

Computer 4
Application B
AMS Interface
Application Message Receiver (AMR)
Application Process 1
Application Process 2
Application Process n
Message Application Interface
Business Functions
BF1
BF2
BF3
BF4
BFn
Step 7: The Business Function activates.
Step 8: The Business Function reads the attributes from the request message.
Database Access Functions
Database
Database Management System
Physical Database

Figure 6

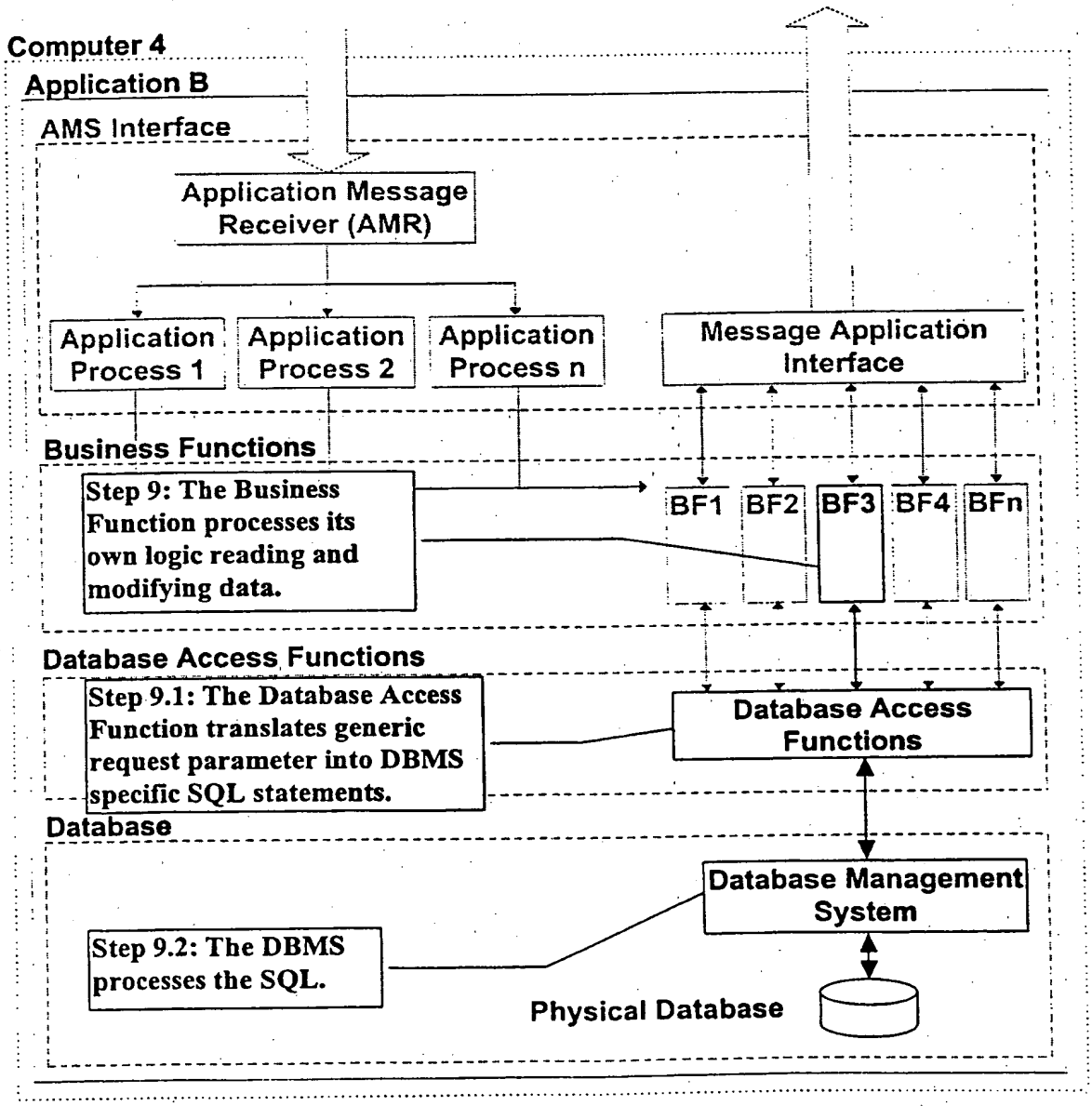

Computer 4
Application B
AMS Interface
Application Message Receiver (AMR)
Application Process 1
Application Process 2
Application Process n
Message Application Interface
Business Functions
Step 9: The Business Function processes its own logic reading and modifying data.
BF1
BF2
BF3
BF4
BFn
Database Access Functions
Step 9.1: The Database Access Function translates generic request parameter into DBMS specific SQL statements.
Database Access Functions
Database
Database Management System
Step 9.2: The DBMS processes the SQL.
Physical Database

Figure 7

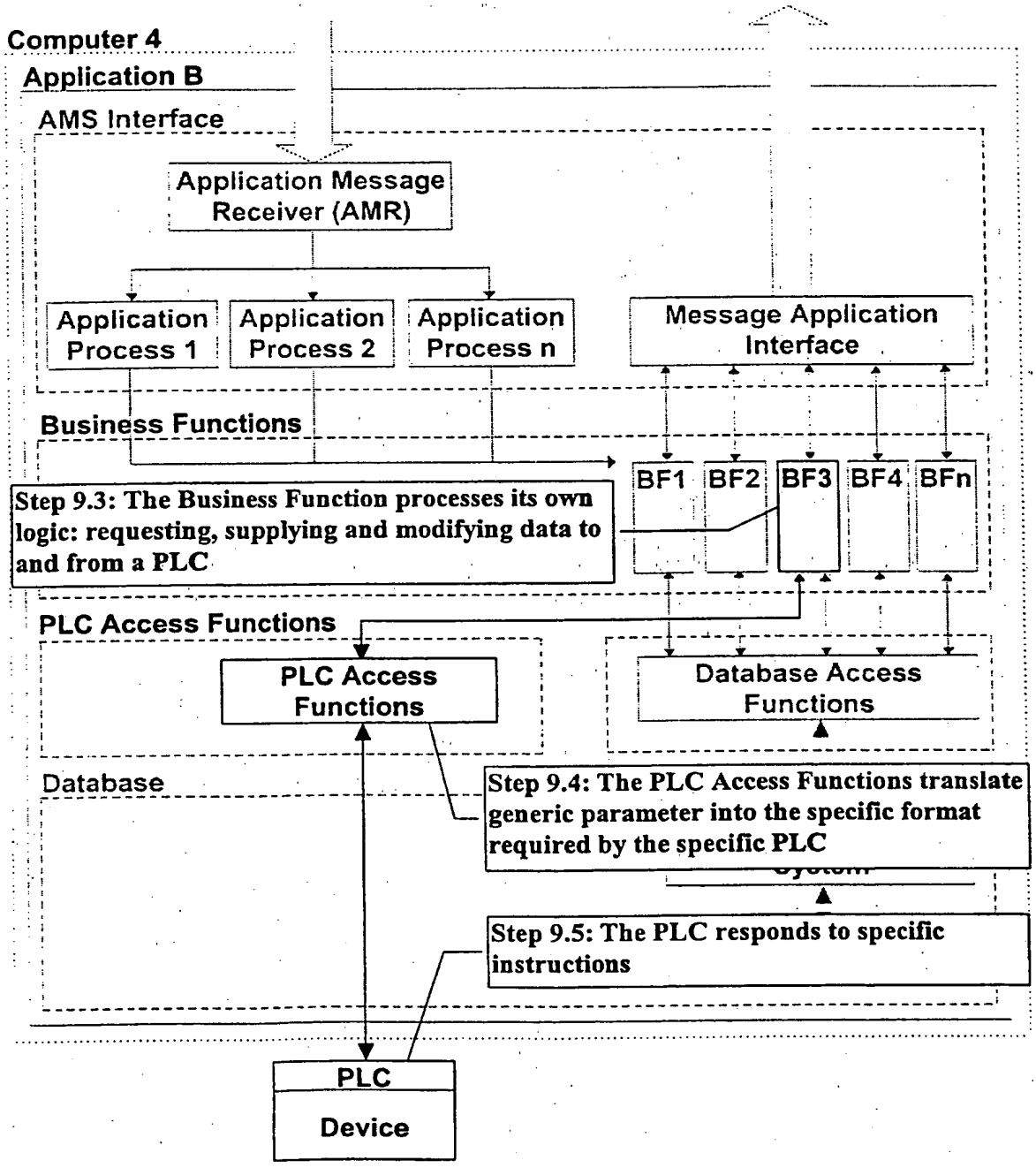

Computer 4
Application B
AMS Interface
Application Message Receiver (AMR)
Application Process 1
Application Process 2
Application Process n
Message Application Interface
Business Functions
BF1
BF2
BF3
BF4
BFn
Step 9.3: The Business Function processes its own logic: requesting, supplying and modifying data to and from a PLC
PLC Access Functions
PLC Access Functions
Database Access Functions
Database
Step 9.4: The PLC Access Functions translate generic parameter into the specific format required by the specific PLC
Step 9.5: The PLC responds to specific instructions
PLC
Device

Figure 7.1

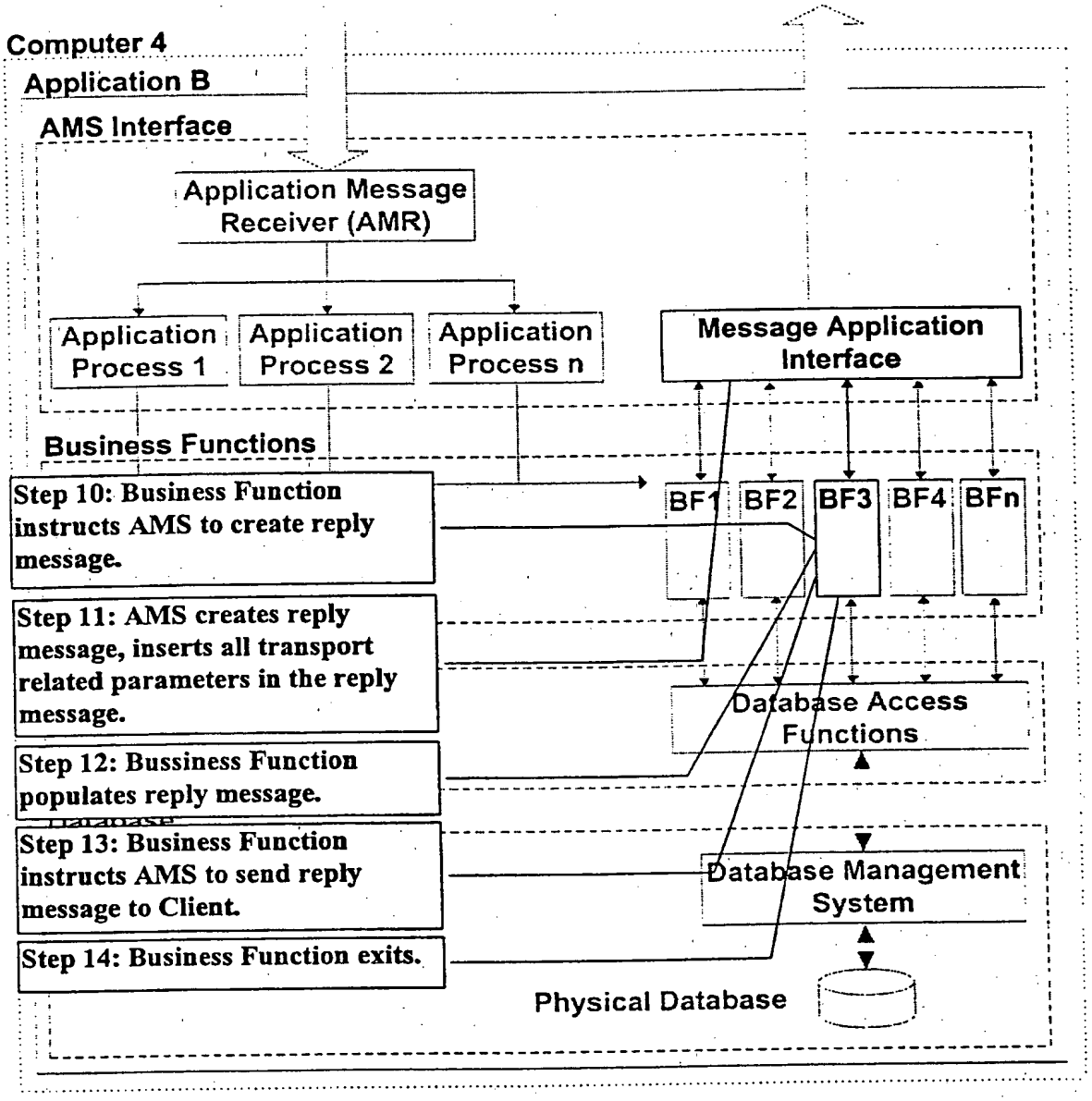

Computer 4
Application B
AMS Interface
Application Message Receiver (AMR)
Application Process 1
Application Process 2
Application Process n
Message Application Interface
Business Functions
Step 10: Business Function instructs AMS to create reply message.
Step 11: AMS creates reply message, inserts all transport related parameters in the reply message.
Step 12: Bussiness Function populates reply message.
Step 13: Business Function instructs AMS to send reply message to Client.
Step 14: Business Function exits.
BF1
BF2
BF3
BF4
BFn
Database Access Functions
Database Management System
Physical Database

Figure 8

10/10

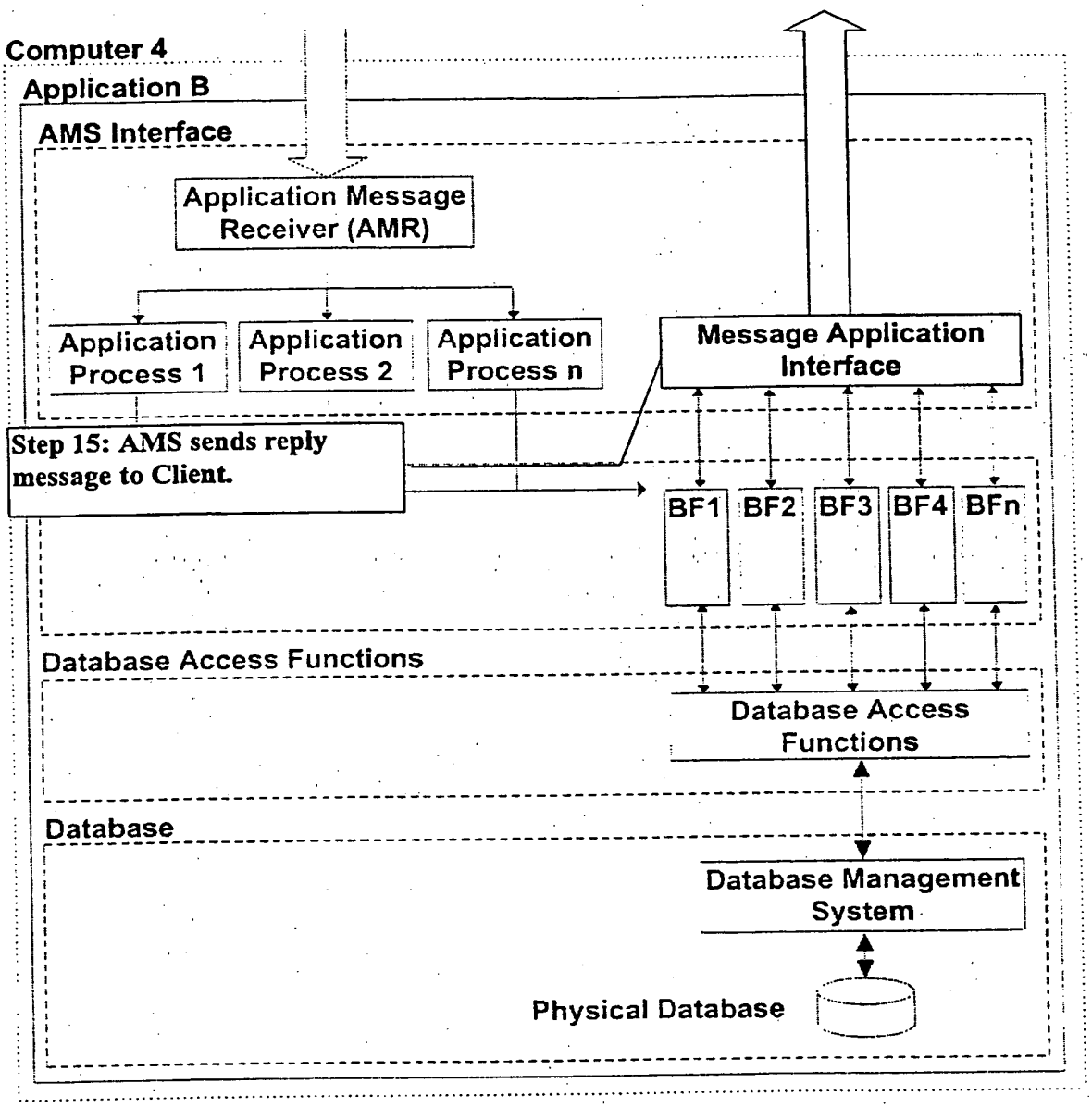


Figure 9